Name: Tushar

Ad No. : 18SCSE1010487

En No. : 18021011715

Batch : 3

Aim : Write a program to demonstrate the working of the Logistic Regression. Use an appropriate data set the implementation.

Theory :

Introduction to Logistic Regression
Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Logistic Regression Assumptions
Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other.
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

Regression Models
- Binary Logistic Regression Model − The simplest form of logistic regression is binary or binomial logistic regression in which the target or dependent variable can have only 2 possible types either 1 or 0.
- Multinomial Logistic Regression Model − Another useful form of logistic regression is multinomial logistic regression in which the target or dependent variable can

have 3 or more possible ***unordered*** types i.e. the types having no quantitative significance.

Binary Logistic Regression model

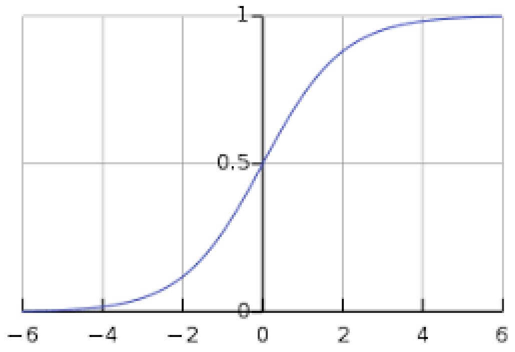
The simplest form of logistic regression is binary or binomial logistic regression in which the target or dependent variable can have only 2 possible types either 1 or 0. It allows us to model a relationship between multiple predictor variables and a binary/binomial target variable. In case of logistic regression, the linear function is basically used as an input to another function such as g in the following relation −

hθ(x)=g(θTx)where0≤hθ≤1

Here, g is the logistic or sigmoid function which can be given as follows −

g(z)=11+e−zwherez=θTx

To sigmoid curve can be represented with the help of following graph. We can see the values of y-axis lie between 0 and 1 and crosses the axis at 0.5.



The classes can be divided into positive or negative. The output comes under the probability of positive class if it lies between 0 and 1. For our implementation, we are interpreting the output of hypothesis function as positive if it is ≥0.5, otherwise negative.

We also need to define a loss function to measure how well the algorithm performs using the weights on functions, represented by theta as follows −

$h=g(X\theta)$

J(θ)=1m.(−yTlog(h)−(1−y)Tlog(1−h))

Now, after defining the loss function our prime goal is to minimize the loss function. It can be done with the help of fitting the weights which means by increasing or decreasing the weights. With the help of derivatives of the loss function w.r.t each weight, we would be able to know what parameters should have high weight and what should have smaller weight.

The following gradient descent equation tells us how loss would change if we modified the parameters –

δJ(θ)δθj=1mXT(g(Xθ)−y)

The Iris Dataset

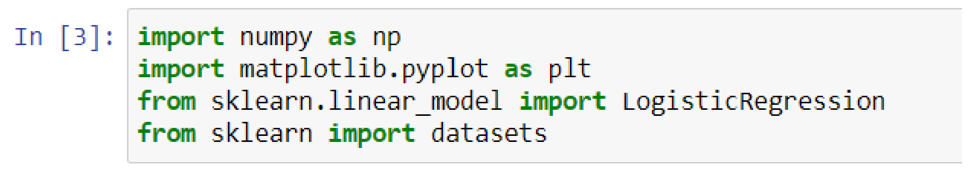
This data sets consists of 3 different types of irises' (Setosa, Versicolour, and Virginica) petal and sepal length, stored in a 150x4 numpy.ndarray

The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width.

Implementation :

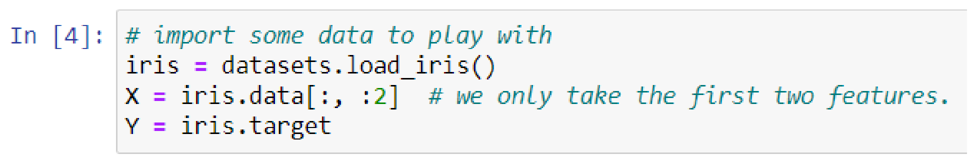
Now we will implement the above concept of binomial logistic regression in Python. For this purpose, we are using a multivariate flower dataset named 'iris' which have 3 classes of 50 instances each, but we will be using the first two feature columns. Every class represents a type of iris flower.

First, we need to import the necessary libraries as follows –



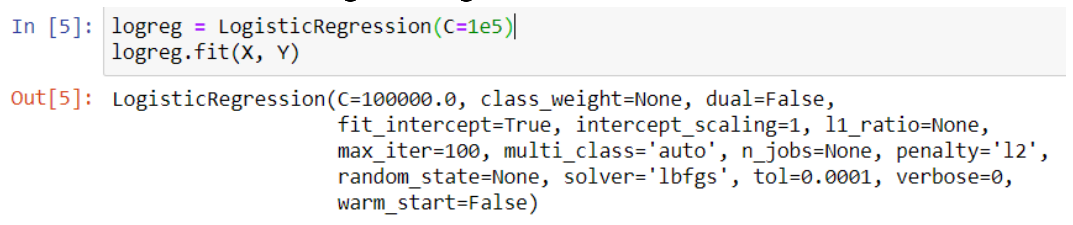
```
In [3]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.linear_model import LogisticRegression
        from sklearn import datasets
```

Next, load the iris dataset as follows –



```
In [4]: # import some data to play with
        iris = datasets.load_iris()
        X = iris.data[:, :2]  # we only take the first two features.
        Y = iris.target
```

Create an instance of Logistic Regression Classifier and fit the data.



```
In [5]: logreg = LogisticRegression(C=1e5)
        logreg.fit(X, Y)

Out[5]: LogisticRegression(C=100000.0, class_weight=None, dual=False,
                           fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                           max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                           random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                           warm_start=False)
```

Plot the decision boundary. For that, we will assign a color to each point in the mesh [x_min, x_max]x[y_min, y_max].

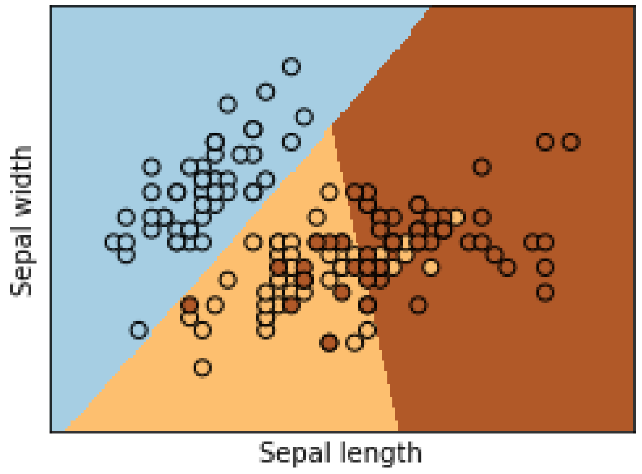
```
In [6]: x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
        y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
        h = .02  # step size in the mesh
        xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
        Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
```

Put the result into a color plot and Plot also the training points

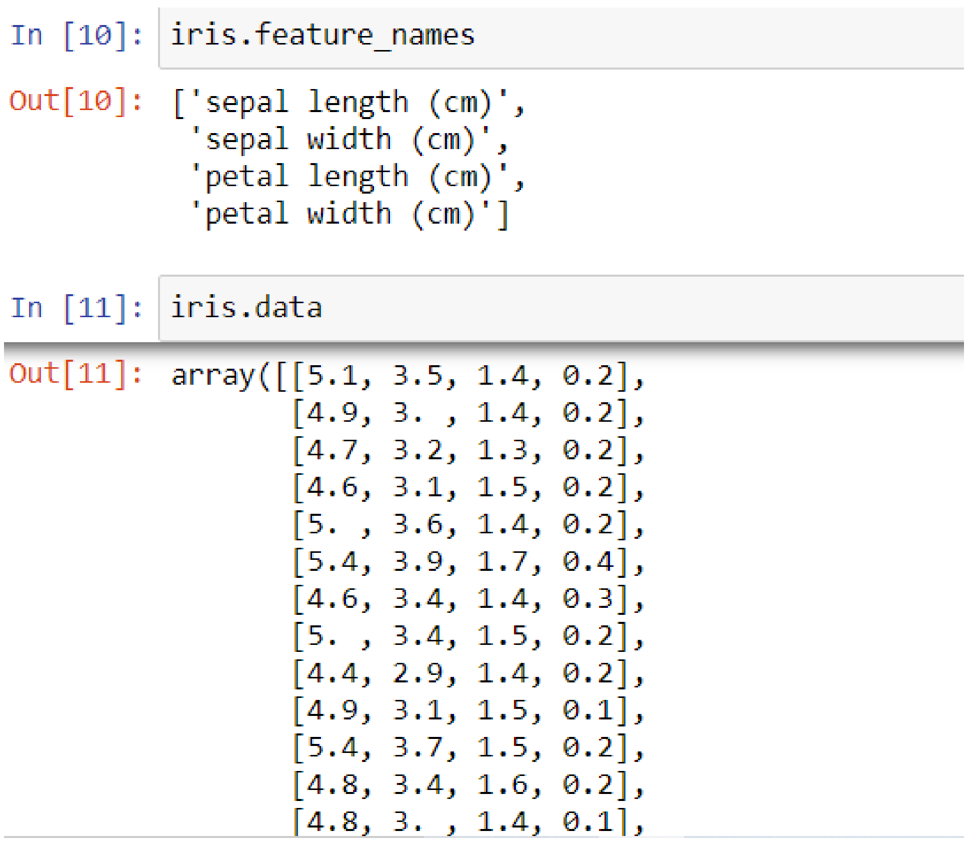
```
In [9]: # Put the result into a color plot
        Z = Z.reshape(xx.shape)
        plt.figure(1, figsize=(4, 3))
        plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
        # Plot also the training points
        plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
        plt.xlabel('Sepal length')
        plt.ylabel('Sepal width')

        plt.xlim(xx.min(), xx.max())
        plt.ylim(yy.min(), yy.max())
        plt.xticks(())
        plt.yticks(())

        plt.show()
```



Show Iris Features and Data

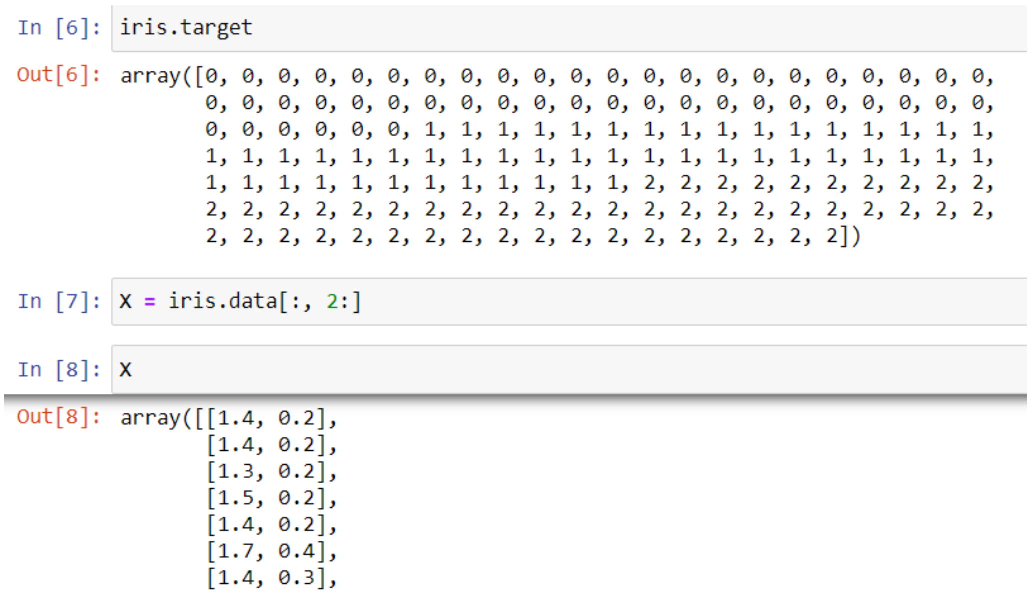


```
In [10]: iris.feature_names

Out[10]: ['sepal length (cm)',
          'sepal width (cm)',
          'petal length (cm)',
          'petal width (cm)']

In [11]: iris.data

Out[11]: array([[5.1, 3.5, 1.4, 0.2],
                [4.9, 3. , 1.4, 0.2],
                [4.7, 3.2, 1.3, 0.2],
                [4.6, 3.1, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.4, 3.9, 1.7, 0.4],
                [4.6, 3.4, 1.4, 0.3],
                [5. , 3.4, 1.5, 0.2],
                [4.4, 2.9, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.1],
                [5.4, 3.7, 1.5, 0.2],
                [4.8, 3.4, 1.6, 0.2],
                [4.8, 3. , 1.4, 0.1],
```



```
In [6]: iris.target

Out[6]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

In [7]: X = iris.data[:, 2:]

In [8]: X

Out[8]: array([[1.4, 0.2],
               [1.4, 0.2],
               [1.3, 0.2],
               [1.5, 0.2],
               [1.4, 0.2],
               [1.7, 0.4],
               [1.4, 0.3],
```

Now we are gonna add libraries for test datasets

In [10]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
import pandas as pd
%matplotlib inline
```

In [11]:
```python
dataset = sns.load_dataset("iris")
```

In [12]:
```python
dataset
```

Out[12]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

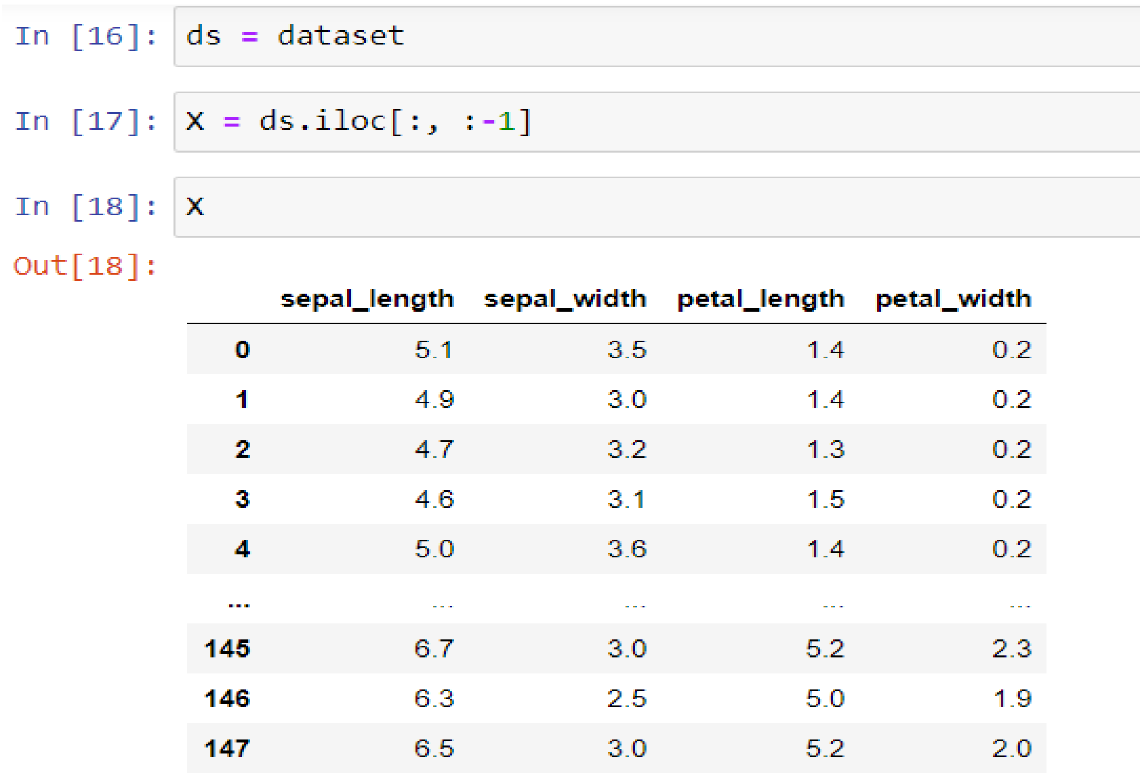
In [13]:
```python
dataset_2 = datasets.load_iris
```

In [14]:
```python
dataset_2
```

Out[14]:
```
<function sklearn.datasets._base.load_iris(return_X_y=False)>
```

In [15]:
```python
print(dataset_2)
```

```
<function load_iris at 0x0000021A69EDA3A8>
```



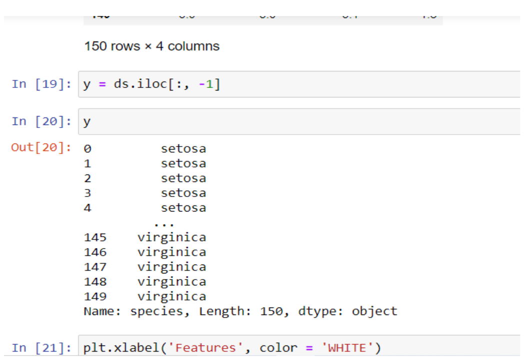
```
In [16]: ds = dataset
```

```
In [17]: X = ds.iloc[:, :-1]
```

```
In [18]: X
```

Out[18]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0



150 rows × 4 columns

```
In [19]: y = ds.iloc[:, -1]
```

```
In [20]: y
```

```
Out[20]: 0         setosa
         1         setosa
         2         setosa
         3         setosa
         4         setosa
                    ...
         145    virginica
         146    virginica
         147    virginica
         148    virginica
         149    virginica
         Name: species, Length: 150, dtype: object
```

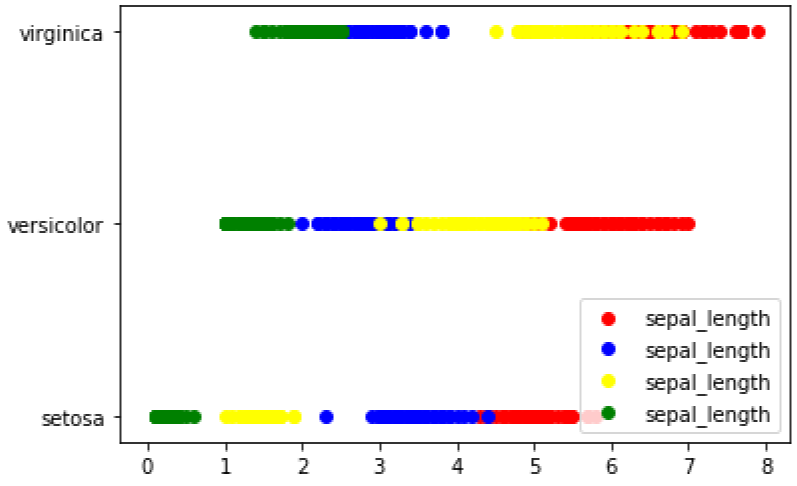
```
In [21]: plt.xlabel('Features', color = 'WHITE')
In [21]: plt.xlabel('Features', color = 'WHITE')
         plt.ylabel('Species', color = 'WHITE')
         pltX = ds.loc[:, 'sepal_length']
         pltY = ds.loc[:, 'species']
         plt.scatter(pltX, pltY, color = 'red', Label = 'sepal_length')

         pltX = ds.loc[:, 'sepal_width']
         pltY = ds.loc[:, 'species']
         plt.scatter(pltX, pltY, color = 'blue', Label = 'sepal_length')

         pltX = ds.loc[:, 'petal_length']
         pltY = ds.loc[:, 'species']
         plt.scatter(pltX, pltY, color = 'yellow', Label = 'sepal_length')

         pltX = ds.loc[:, 'petal_width']
         pltY = ds.loc[:, 'species']
         plt.scatter(pltX, pltY, color = 'green', Label = 'sepal_length')
         plt.legend()

Out[21]: <matplotlib.legend.Legend at 0x21a6c6f9f08>
```



```
In [22]: x_train, x_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, test_size = 0.2, random_sta
         model = LogisticRegression()
         model.fit(x_train, y_train)

Out[22]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='auto', n_jobs=None, penalty='l2',
                            random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                            warm_start=False)

In [23]: results = model.predict(x_test)

In [24]: results

Out[24]: array(['setosa', 'versicolor', 'virginica', 'virginica', 'versicolor',
                'virginica', 'versicolor', 'versicolor', 'versicolor', 'setosa',
                'versicolor', 'setosa', 'setosa', 'virginica', 'versicolor',
                'virginica', 'virginica', 'virginica', 'versicolor', 'versicolor',
                'virginica', 'virginica', 'versicolor', 'setosa', 'versicolor',
                'setosa', 'setosa', 'virginica', 'setosa', 'versicolor'],
               dtype=object)
```

```
In [25]: print(classification_report(y_test, results))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         8
  versicolor       1.00      1.00      1.00        12
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
In [26]: pltX = y_test
         pltY = results
         plt.scatter(pltX, pltY, color = 'red', Label = "Predictions")

Out[26]: <matplotlib.collections.PathCollection at 0x21a6c6fe5c8>
```

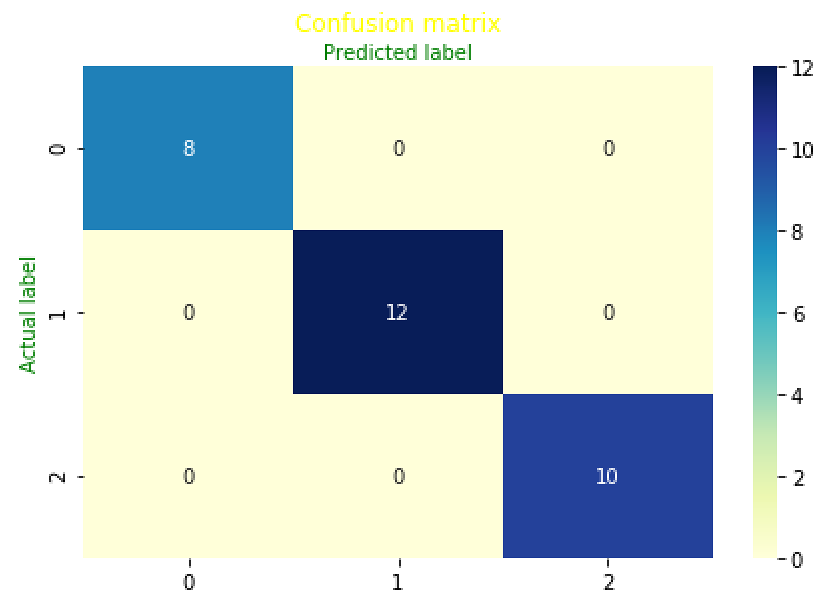
```
In [27]: cnf_matrix = metrics.confusion_matrix(y_test, results)
         cnf_matrix

Out[27]: array([[ 8,  0,  0],
                [ 0, 12,  0],
                [ 0,  0, 10]], dtype=int64)
```

```
In [28]: class_names=[0,1]
         fig, ax = plt.subplots()
         tick_marks = np.arange(len(class_names))
         plt.xticks(tick_marks, class_names)
         plt.yticks(tick_marks, class_names)

         sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
         ax.xaxis.set_label_position("top")
         plt.tight_layout()
         plt.title('Confusion matrix', y=1.1, color = 'YELLOW' )
         plt.ylabel('Actual label', color = 'GREEN')
         plt.xlabel('Predicted label', color = 'GREEN')

Out[28]: Text(0.5, 257.44, 'Predicted label')
```



Result :

In this experiment we are using logistic regression on Iris datasets to classify three types of flowers using iris features namely ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'’] and we have successfully execute it.